

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**DIRECT MEMORY ACCESS (DMA) TRANSFER OF
NETWORK INTERFACE STATISTICS**

Inventor(s): Charles E. Narad

Prepared by:
Robert A. Greenberg



Intel Corporation
HD2-305
77 Reed Road
Hudson, MA 01749

Express Mail Label: EV325527405US

DIRECT MEMORY ACCESS (DMA) TRANSFER OF NETWORK INTERFACE STATISTICS

REFERENCE TO RELATED APPLICATIONS

[0001] This application relates to attorney docket number P17969 entitled "GENERATING PACKETS", filed on the same day as the present application and naming the same inventor.

BACKGROUND

[0002] Networks enable computers and other devices to communicate. For example, networks can carry data representing video, audio, e-mail, and so forth. Typically, data sent across a network is divided into smaller messages known as packets. By analogy, a packet is much like an envelope you drop in a mailbox. A packet typically includes "payload" and a "header". The packet's "payload" is analogous to the letter inside the envelope. The packet's "header" is much like the information written on the envelope itself. The header can include information to help network devices handle the packet appropriately. For example, the header can include an address that identifies the packet's destination. A given packet may travel across many network nodes (e.g., "routers", "bridges" and "switches") before reaching its destination.

[0003] FIG. 1 illustrates an example of components forming a network node. As shown, the node includes a network interface that connects a processor system to a network (shown as a cloud). Typically, an intermediate node, such as a router, will include many different network interfaces. As shown, the interface carries packets traveling between the processor system and the network.

[0004] Network interfaces often compile statistics on their operation such as the number of packets or bytes received or transmitted. For instance, as shown, the interface updates the statistics for packets received (packet “a”) and sent (e.g., packet “b”).

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] FIG. 1 is a flow-diagram illustrating operation of a network interface.

[0006] FIG. 2 is a flow-diagram illustrating Direct Memory Access (DMA) transfer of statistics metering operation of a network interface.

[0007] FIG. 3 is a diagram of a Media Access Controller (MAC).

DETAILED DESCRIPTION

[0008] As described above, in addition to carrying data between a processor system and a network, network interfaces often collect statistics monitoring operation of the interface. For example, such statistics can meter network traffic received or sent by the interface. These statistics can be analyzed by a processor, for example, to monitor performance of the connection. Transferring the statistics for analysis can, however, impose a significant burden on the processor(s). For example, in some systems, the processor(s) may issue a series of requests for statistic values and devote considerable resources to overseeing storage of the values in memory.

[0009] FIG. 2 depicts a scheme that uses a technique known as Direct Memory Access (DMA) to transfer statistics 108 to a memory 104 accessed by processor(s) 106. Briefly, Direct Memory Access permits memory access without involvement of a processor. That is, DMA enables an entity to share access to a memory's 104 address and data busses instead of requiring memory operations to pass through the processor(s) 106. This technique can relieve processors(s) 106 of

the burden of handling the transfer of each statistic value, freeing the processor(s) for other tasks.

To further reduce processor involvement, the statistics transfer can be configured to occur automatically, for example, at periodic intervals or when certain events occur.

[0010] In greater detail, FIG. 2 depicts a network interface 102 that collects network interface statistics 108. For example, a standard called RMON (Internet Engineering Task Force, Request for Comments #3577, Introduction to Remote Monitoring (RMON) Family of MIB Modules, Waldbusser, et al., August 2003) specifies a set of counters that include the number of bytes sent and received, number of packets sent and received, "buckets" of packet size ranges, various network congestion and error conditions, and so forth.

[0011] As shown, the network interface 102 includes a DMA Unit 110 that transfers statistic values 108 to the memory 104. The DMA Unit 110 circuitry may vary depending on the DMA architecture used. For example, the DMA Unit 110 may act as a memory master over a Peripheral Component Interconnect (PCI) bus. Once transferred, the statistics 108 may be accessed by processor(s) 106. The processor(s) 106 can include a Central Processing Unit (CPU) and/or engine(s) of a network processor that aggregates many such processing engines on a single integrated die.

[0012] The interface 102 can be configured to transfer the statistics to a particular location in memory 104. For example, the location may correspond to the location of a data structure mapping a block of the transferred data to different counter values. Potentially, processor(s) 106 may access the data while a DMA transfer is occurring. This may result in some inconsistency in the data accessed. For example, the "packets sent" statistic accessed by the processor(s) 106 may be from a previous DMA transfer while the "packets received" statistic accessed may have just been written by the current DMA transfer. In many cases, this discrepancy is of little

importance as many applications using this data assume that individual counter values are not precisely correlated, instead looking for long-term statistical trends.

[0013] While usually not a requirement, a variety of techniques can provide statistic consistency for applications. For example, the DMA unit 110 and processor(s) 106 may share a mutex (mutual exclusion) lock that enables only one entity 110, 106 to access the memory 104 locations storing the statistics 108 at a time. Alternately, the DMA unit 110 may change the location used to store the statistics 108, creating a series of "snapshots" of the interface's counter values at different times. The processor(s) 106 can then restrict its access, for example, to the most recently completed snapshot.

[0014] The storage of these snapshots can be controlled in a variety of ways. For example, the locations used to store the snapshots may correspond to different elements of a "ring" where, eventually, after completing a circle around the ring, a new snapshot overwrites the oldest. Alternatively, the location of a snapshot may be set to a buffer allocated from a freelist. The buffer may be appended to a linked list after being populated with DMA-ed data. The processor(s) 106 may be restricted from pre-maturely accessing a buffer until the buffer is linked into the list. The data transferred in a snapshot might contain additional information such as a snapshot sequence number or a timestamp indicating the approximate time at which the snapshot was captured.

[0015] The transfer destination locations may be pre-configured in the interface 102. Alternately, the processor(s) 106 may issue commands to the interface 102 that specify the location in memory 104 in which to dump all or a specified subset of the statistics.

[0016] After transferring the statistics 108 data, the interface 102 may signal completion of the transfer to the processor(s). For example, the interface 102 can generate a processor interrupt

signal. Alternatively the interface 102 may indicate in memory 104 that the transfer is complete by indications including flags, ring pointers, linked-list pointers and so forth. The processor(s) 106 might poll or test periodically or at random intervals the locations in memory containing such indications to determine if a transfer has occurred. The processor(s) 106 may instead access the location(s) in memory 104 containing the statistics 108 at a time based on prior knowledge of the time or periodicity of statistics 108 delivery into memory 104 by the interface 102.

[0017] After, or even during a transfer, the processor(s) 106 can perform a variety of operations. For example, the processor(s) 106 can determine counter changes ("deltas") since the last transfer. The processor(s) 106 can then use these deltas, for example, to update its master tabulation of counter values (not shown) or to trigger an alarm if a predetermined threshold is reached.

[0018] Potentially, the processor(s) 106 may include a cache (not shown). For example, the cache may store statistic values recently accessed by the processor(s) 106. While speeding access to this data for the processor(s) 106, the cache can become outdated. For example, a new set of statistic values 108 may be transferred after previously transferred values were stored in the cache. To prevent cached statistic values from going stale, memory locations storing transferred statistics 108 may be mapped/marked as non-cacheable. Alternately, the cached values may be cleared or updated upon receipt of updating statistic data utilizing a cache coherence mechanism such as write-invalidate or write-update. Potentially, the statistic values 108 may be pushed directly into the processor(s) 106 cache, and either mirrored in memory 104 or only to be written to memory 104 when victimized from the cache.

[0019] The network interface 102 may include a variety of hardware and/or software components. For example, the network interface 102 may include a PHY (physical layer device) that performs analog-to-digital conversion on received signals (e.g., wire, optic, or wireless analog signals). The PHY may feed a framer that groups the bits output by the PHY into frame packets, for example, by identifying bits signaling the start and end of a frame packet. The framer may also perform other operations such as bit/character stuffing and unstuffing, checksum computation and verification, packet de/en-capsulation, serialization/deserialization and so forth. The framer operations and packet format depend on the framing technology supported (e.g., Synchronous Optical NETwork (SONET), Ethernet, High-Level Data Link Control (HDLC), and Point-to-Point Protocol (PPP)). The network interface 102 may include other components such as a Transmission Control Protocol (TCP) Offload Engine (TOE) that offloads TCP operations (e.g., ACK generation, segment reassembly, and so forth) from the processor(s) 106.

[0020] FIG. 3 depicts an example of an Ethernet Media Access Controller (MAC) 120 framer that may be found in a network interface 102 using DMA to transfer statistic values. As shown, the framer 120 includes circuitry to perform receive (Rx) 122 and transmit (Tx) 134 framing operations on in-bound and out-bound data, respectively. Both sets of circuitry 122, 134 can update statistic values 124 being monitored. As shown, the framer 120 also includes a DMA unit 128 that can transfer received packets to memory as requested by the Rx 122 circuitry. The DMA unit 128 is also operationally coupled to transfer control circuitry 126 to initiate DMA transfer of statistic values 124.

[0021] Operations performed by the transfer control circuitry 126 can be configured in a variety of ways. For example, the circuitry 126 may be configured to select some subset of monitored statistic values to transfer. Similarly, the circuitry 126 may be configured to automatically

transfer some or all statistic values 124 at particular intervals or when particular values reach pre-configured thresholds. Potentially, these configuration options may be combined to specify transfer of different sets of statistics at different times (e.g., transfer "send" statistics at time interval "1" and "receive" statistics at time interval "2").

[0022] The framer 120 may also be configured to select a particular mechanism used to maintain counter values 124. Briefly, a counter is much like a car's odometer - when the counter reaches its maximum value, it rolls-over back to zero. The interface 102 may be configured to let the counter's "free run". Alternately, the interface 102 may be configured to zero the counters after the statistics 108 are transferred to memory 104. Such counters would reflect a change since the last statistics transfer. To prevent ambiguity in the data, the framer can be configured to transfer statistics 108 at a frequency great enough to prevent counter wrap-around.

[0023] As shown, the interface framer 120 can be configured using a variety of mechanisms. For example, the framer can be configured by different registers (not shown) that the processor(s) can access. For instance, a "1" stored in bit-1 of a register may select the "packets sent" statistic for transfer while other register bits identify a time interval for transferring the statistic. Alternately, the framer 120 may include circuitry 136 to intercept packets, for example, traveling along the interface's 120 transmit path (e.g., the path leading to the network) or receive path (e.g., the path leading to the processor(s) 106). For instance, the processor(s) 106, or other entity, may construct a packet having characteristics identifying the packet as one carrying data to configure the interface 120 instead of one to be transmitted over the network. As an example, such a packet may include a source and destination address having some preset value(s). The payload of such packets may include data identifying, for example, statistics to transfer, intervals or specific times to transfer data, a command for an immediate transfer, a schedule of differing

transfers, and/or threshold statistic values of events that trigger a transfer. The payload may further include data identifying the location(s) in memory 104 in which to place statistics. The intercept circuitry 136 may use this packet data to program operation of the transfer control circuitry 126. For example, the intercept 136 circuitry may use this data to program a timer (not shown) that initiates transfer. The mechanisms (e.g., registers or packets) used to configure the framer may also be used to make a "one-shot" transfer request.

[0024] The preceding description frequently used the term "packet" to refer to a frame. However, the term packet also describes Transmission Control Protocol (TCP) segments, Internet Protocol (IP) datagrams, Asynchronous Transfer Mode (ATM) cells, and so forth.

[0025] The term circuitry as used herein includes hardwired circuitry, digital circuitry, analog circuitry, programmable circuitry, and so forth. The programmable circuitry may operate on computer programs. For example, the transfer control 126 and intercept 136 circuitry may be implemented by a microcontroller programmed to perform operations described above. Such programs may be coded in a high level procedural or object oriented programming language. However, the program(s) can be implemented in assembly or machine language if desired. The language may be compiled or interpreted.

[0026] Techniques described above may be used in a wide variety of networking environments. Further, techniques described above may be incorporated into a variety of components such as a Network Interface Controller (NIC) chip and/or card or included in a motherboard chipset or network processing unit (NPU). These techniques may also be implemented in a router or switch line card.

[0027] Other embodiments are within the scope of the following claims.